```java
// ase2operation.java
// Copyright (c) 2003. Sybase, Inc. All Rights Reserved.
//
//Confidential property of Sybase, Inc.
//Copyright 1987, 2003
//Sybase, Inc.  All rights reserved.
//Unpublished rights reserved under U.S. copyright laws.
//
//
package com.sybase.ase.ws.server;
/**
 * FileName:   ase2operation.java
 * PackageName: com.sybase.ase.ws.server
 *
 * Description:
 * This represents the translation of a WSDL operation or a web method
 * into ASE speak.
 *
 */
import java.util.Vector;
import java.util.List;
import java.util.Iterator;
import com.sybase.ase.ws.util.Utility;
import com.sybase.ase.ws.util.Globals;
import com.sybase.ase.ws.util.WSParser;
import com.sybase.jdbc2.tds.SrvDataFormat;
import com.sybase.jdbc2.tds.TdsConst;
import org.apache.axis.wsdl.symbolTable.SymbolTable;
import org.apache.axis.enum.Style;
import javax.wsdl.extensions.ExtensibilityElement;
import javax.xml.namespace.QName;
import com.ibm.wsdl.extensions.soap.SOAPConstants;
import com.ibm.wsdl.extensions.soap.SOAPOperationImpl;
import javax.wsdl.Operation;
import javax.wsdl.Input;
import javax.wsdl.Output;
import javax.wsdl.Message;
import javax.wsdl.Part;
import javax.wsdl.BindingOperation;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.net.URLClassLoader;
import java.net.URL;
import org.apache.axis.client.Service;
import org.apache.axis.client.Call;
import org.apache.axis.message.SOAPBodyElement;
public class ase2operation
{
    // Also used as the table name in "create existing table" statement.
    String   _aseRPCName;
```

```java
String    _operationName;
Input   _Input;
Output   _Output;
Operation _Operation;
Vector  _InputParameters;
Vector  _OutputParameters;
ase2service _serviceEntry;
ase2wsdl    _wsdlEntry;
BindingOperation _bindingOp;
SOAPOperationImpl _operationImpl;
ase2parameter _returnParameter;
public ase2parameter getReturnParmeter ()
{
    return (_returnParameter);
}
private Object[] fillArgs (Object[] args)
{
    if (_OutputParameters == null)
    {
        return (args);
    }
    if (_OutputParameters.size () == 0)
    {
        return (args);
    }
    if (_OutputParameters.size () == 1 && _returnParameter != null)
    {
        return (args);
    }
    int totalSize = args.length + _OutputParameters.size ();
    Object[] allArgs = new Object[totalSize];
    int count;
    for (count = 0; count < args.length; count++)
    {
        allArgs[count] = args[count];
    }
    ase2parameter aParm = null;
    for (count = 0; count < _OutputParameters.size (); count++)
    {
        try
        {
            aParm = (ase2parameter) _OutputParameters.get (count);
            Class tempClass = Class.forName (aParm.getJavaType ());
            allArgs[count + args.length] = tempClass.newInstance ();
        }
        catch (ClassNotFoundException err)
        {
            Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
            + err.getMessage ());
        }
```

```java
      catch (SecurityException err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (IllegalArgumentException err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (NullPointerException err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (IllegalAccessException err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (InstantiationException err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (ExceptionInInitializerError err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
      catch (LinkageError err)
      {
          Globals.xmlLogger.fatal (Utility.getMessage ("outputparamex ")
          + err.getMessage ());
      }
    }
    return (allArgs);
}
private Object invokeWebMethodStub (Object[] args)
    throws ClassNotFoundException, InvocationTargetException, IllegalArgumentException,
    IllegalAccessException, InstantiationException
{
    Object toReturn = null;
    URLClassLoader jarLoader = _wsdlEntry.getJarLoader ();
    String locatorClassName = _serviceEntry.getServiceJavaName ()
    + "Locator";
    Class locatorClass = Class.forName (locatorClassName, true, jarLoader);
    Object locatorObject = locatorClass.newInstance ();
    // Execute getPort method.
    String getStubMethodName = "get" + _serviceEntry.getPortName ();
```

```java
      Method getStubMethod = null;
      Method[] allMethods = locatorClass.getMethods ();
      for (int ii = 0; ii < allMethods.length; ii++)
      {
         if (allMethods[ii].getName ().compareToIgnoreCase (getStubMethodName)
         == 0)
         {
            getStubMethod = allMethods[ii];
            break;
         }
      }
      Object stubObject = getStubMethod.invoke (locatorObject, null);
      Class stubClass = stubObject.getClass ();
      // Find webmethod of name _Operation.getName ();
      Method webMethod = null;
      allMethods = stubClass.getMethods ();
      for (int ii = 0; ii < allMethods.length; ii++)
      {
         if (allMethods[ii].getName ().compareToIgnoreCase (_Operation.getName ())
         == 0)
         {
            webMethod = allMethods[ii];
            break;
         }
      }
      // Invoke web method.
      toReturn = webMethod.invoke (stubObject, args);
      return (toReturn);
}
private opReturn invokeRPC (Object[] args) throws Exception
{
      Object webMethodReturn = null;
      opReturn toReturn = null;
      if (args != null)
      {
         Object[] allArgs = fillArgs (args);
         int numOutputArgs = allArgs.length - args.length;
         int numInputArgs = args.length;
         webMethodReturn = invokeWebMethodStub (args);
         int numParams = 0;
         toReturn = new opReturn ();
         if (_returnParameter != null)
         {
            numParams = 1;
            toReturn.desc = new SrvDataFormat[numOutputArgs + 1];
            toReturn.data = new Object[1][numOutputArgs + 1];
         }
         else
         {
            toReturn.desc = new SrvDataFormat[numOutputArgs];
```

```
            toReturn.data = new Object[1][numOutputArgs];
        }
        int tempParamIndex = 0;
        ase2parameter tempParam;
        int count = 0;
        if (_OutputParameters != null)
        {
            numParams += _OutputParameters.size ();
            for (; count < _OutputParameters.size (); count++)
            {
                tempParam = (ase2parameter) _OutputParameters.get (count);
                toReturn.desc[count] = tempParam.getSrvDataFormat ();
                toReturn.data[0][count] = tempParam.convert (allArgs[numInputArgs + count]);
                tempParamIndex++;
            }
        }
        // Now handle the return arg, if one exists.
        if (_returnParameter != null)
        {
            toReturn.desc[tempParamIndex] = _returnParameter.getSrvDataFormat ();
            toReturn.data[0][tempParamIndex] = _returnParameter.convert (webMethodReturn);
        }
    }
    else
    {// We have to have some args or we bail out.
        // This may turn into a hack for the optimizer.
    }
    return (toReturn);
}
private opReturn invokeDocument (Object[] args) throws Exception
{
    opReturn toReturn = null;
    Service service = new Service ();
    Call call = (Call) service.createCall ();
    call.setTargetEndpointAddress (new URL (_serviceEntry._locationURI));
    call.setSOAPActionURI (_operationImpl.getSoapActionURI ());
    call.setPortName (_serviceEntry._portTypeName);
    WSParser parser = new WSParser ();
    SOAPBodyElement[] sbElements = parser.getSoapBody ((String) args[0]);
    Vector elems = null;
    if (sbElements.length != 0)
    {
        elems = (Vector) call.invoke (sbElements);
        toReturn = new opReturn ();
        toReturn.desc = new SrvDataFormat[1];
        toReturn.data = new Object[1][elems.size ()];
        StringBuffer allData = new StringBuffer ();
        SOAPBodyElement tempBodyElement = null;
        StringBuffer rowData;
        int ii = 0;
```

```java
        if (elems != null)
        {
            Iterator elemsI = elems.iterator ();
            while (elemsI.hasNext ())
            {
                tempBodyElement = (SOAPBodyElement) elemsI.next ();
                // Note that we only support UTF-8 character set. No conversions at all.
                rowData = new StringBuffer ("<?xml version=\"1.0\" encoding=\"UTF-8\" ?>");
                rowData.append (tempBodyElement.toString ());
                toReturn.data[0][ii] = rowData.toString ().getBytes ();
                ii++;
                rowData = null;
                // allData.append(tempBodyElement.toString());
            }
            toReturn.desc[0] = new SrvDataFormat ("outxml",
            TdsConst.LONGBINARY, TdsConst.ROW_UPDATABLE, 15000, null);
        }
    }
    return (toReturn);
}
public opReturn invoke (Object[] args) throws Exception
{
    opReturn toReturn = null;
    if (_serviceEntry._bindingStyle == Style.RPC)
    {
        toReturn = invokeRPC (args);
    }
    else if (_serviceEntry._bindingStyle == Style.DOCUMENT)
    {
        toReturn = invokeDocument (args);
    }
    return (toReturn);
}
public String getOperationName ()
{
    return _operationName;
}
public String getAseRPCName ()
{
    return _aseRPCName;
}
public Vector getInputParameters ()
{
    return _InputParameters;
}
public Vector getOutputParameters ()
{
    return _OutputParameters;
}
public ase2operation (ase2service serviceEntry, ase2wsdl wsdlEntry, Operation anOperation, SymbolTable
symTable) throws MappingException
```

```
{
    _serviceEntry = serviceEntry;
    _wsdlEntry = wsdlEntry;
    _operationName = anOperation.getName ();
    _aseRPCName = aseutil.mapIdentifier (anOperation.getName ());
    _Operation = anOperation;
    _Input = anOperation.getInput ();
    _Output = anOperation.getOutput ();
    _bindingOp = serviceEntry._binding.getBindingOperation (_Operation.getName (),
    _Input.getName (), _Output.getName ());
    List extensionList = _bindingOp.getExtensibilityElements ();
    Iterator extensionListI = extensionList.iterator ();
    while (extensionListI.hasNext ())
    {
        ExtensibilityElement eElement = (ExtensibilityElement) extensionListI.next ();
        QName eType = eElement.getElementType ();
        if (eType.equals (SOAPConstants.Q_ELEM_SOAP_OPERATION))
        {
            _operationImpl = (SOAPOperationImpl) eElement;
        }
    }
    if (serviceEntry._bindingStyle == Style.DOCUMENT)
    {// We skip all processing here as the input and output
        // are XML.
    }
    else
    {
        Message inMessage = _Input.getMessage ();
        Message outMessage = _Output.getMessage ();
        ase2parameter aseParm = null;
        List lPart = inMessage.getOrderedParts (null);
        Iterator iPart = lPart.iterator ();
        while (iPart.hasNext ())
        {
            Part aPart = (Part) iPart.next ();
            aseParm = new ase2parameter (aPart, symTable);
            if (_InputParameters == null)
            {
                _InputParameters = new Vector ();
            }
            _InputParameters.add (aseParm);
        }
        lPart = outMessage.getOrderedParts (null);
        iPart = lPart.iterator ();
        while (iPart.hasNext ())
        {
            Part aPart = (Part) iPart.next ();
            aseParm = new ase2parameter (aPart, symTable);
            if (aseParm.isReturn ())
            {
```

```java
                _returnParameter = aseParm;
            }
            else
            {
                if (_OutputParameters == null)
                {
                    _OutputParameters = new Vector ();
                }
                _OutputParameters.add (aseParm);
            }
        }
    }
}
// ase2service.java
// Copyright (c) 2003. Sybase, Inc. All Rights Reserved.
//
//Confidential property of Sybase, Inc.
//Copyright 1987, 2003
//Sybase, Inc.  All rights reserved.
//Unpublished rights reserved under U.S. copyright laws.
//
//
package com.sybase.ase.ws.server;
/**
 * FileName:    ase2service.java
 * PackageName: com.sybase.ase.ws.server
 *
 * Description:
 * This class represents the conversion of a WSDL service into
 * ASE.
 *
 */
import java.util.Vector;
import java.util.Iterator;
import java.util.Map;
import java.util.List;
import org.apache.axis.wsdl.symbolTable.SymbolTable;
import org.apache.axis.wsdl.symbolTable.ServiceEntry;
import org.apache.axis.wsdl.symbolTable.BindingEntry;
import org.apache.axis.enum.Style;
import javax.wsdl.Service;
import javax.wsdl.Port;
import javax.wsdl.Binding;
import javax.wsdl.PortType;
import javax.wsdl.Operation;
import javax.wsdl.extensions.ExtensibilityElement;
import javax.xml.namespace.QName;
import com.ibm.wsdl.extensions.soap.SOAPConstants;
import com.ibm.wsdl.extensions.soap.SOAPAddressImpl;
```

```java
import com.sybase.ase.ws.server.aseutil;
import com.sybase.ase.ws.util.Globals;
import com.sybase.ase.ws.util.LogConstants;
public class ase2service implements LogConstants
{
    String  _aseOwnerName;
    String  _serviceName;
    Vector  _operations;
    Service _service;
    String  _serviceJavaName;
    ase2wsdl _wsdlEntry;
    String  _portName;
    Style   _bindingStyle;
    String  _locationURI;
    Binding _binding;
    QName   _portTypeName;
    public Vector getOperations ()
    {
        return (_operations);
    }
    public String getAseOwnerName ()
    {
        return _aseOwnerName;
    }
    public String getServiceName ()
    {
        return _serviceName;
    }
    public String getServiceJavaName ()
    {
        return _serviceJavaName;
    }
    public String getPortName ()
    {
        return _portName;
    }
    public ase2service (ase2wsdl wsdlEntry, ServiceEntry sEntry, SymbolTable symTable)
    {
        _wsdlEntry = wsdlEntry;
        _service = sEntry.getService ();
        _serviceJavaName = sEntry.getName ();
        _serviceName = aseutil.mapIdentifier (_service.getQName ().getLocalPart ());
        _aseOwnerName = aseutil.mapIdentifier (_service.getQName ().getLocalPart ());
        Map portMap = _service.getPorts ();
        Iterator portIterator = portMap.values ().iterator ();
        while (portIterator.hasNext ())
        {
            Port p = (Port) portIterator.next ();
            _portName = p.getName ();
            _binding = p.getBinding ();
```

```java
        BindingEntry be = symTable.getBindingEntry (_binding.getQName ());
        if (be.getBindingType () != BindingEntry.TYPE_SOAP)
        {
            // We got a binding that's not SOAP, so we
            // skip it.
            continue;
        }
        // Now we are guaranteed to have SOAP extensions, so we
        // go get them.
        List extensionList = p.getExtensibilityElements ();
        Iterator extensionListI = extensionList.iterator ();
        while (extensionListI.hasNext ())
        {
            ExtensibilityElement eElement = (ExtensibilityElement) extensionListI.next ();
            QName eType = eElement.getElementType ();
            if (eType.equals (SOAPConstants.Q_ELEM_SOAP_ADDRESS))
            {
                SOAPAddressImpl addrImpl = (SOAPAddressImpl) eElement;
                _locationURI = addrImpl.getLocationURI ();
            }
        }
        _bindingStyle = be.getBindingStyle ();
        PortType pType = _binding.getPortType ();
        _portTypeName = pType.getQName ();
        List pList = pType.getOperations ();
        Iterator pListI = pList.iterator ();
        while (pListI.hasNext ())
        {
            Operation anOperation = (Operation) pListI.next ();
            try
            {
                ase2operation aseOp = new ase2operation (this, _wsdlEntry,
                anOperation, symTable);
                if (_operations == null)
                {
                    _operations = new Vector ();
                }
                _operations.add (aseOp);
            }
            catch (MappingException err)
            {
                Globals.xmlLogger.fatal ("Skipped an operation");
            }
        }
    }
}
public String genCIS (String service)
{
    StringBuffer cisCommand = new StringBuffer ();
    Iterator iOperations = _operations.iterator ();
```

```
ase2operation aOperation = null;
StringBuffer tempCmd = null;
while (iOperations.hasNext ())
{
    aOperation = (ase2operation) iOperations.next ();
    if (aOperation != null)
    {
        tempCmd = new StringBuffer ("drop table ");
        tempCmd.append (aOperation.getAseRPCName ());
        tempCmd.append ("\n");
        tempCmd.append ("create existing table ");
        tempCmd.append (aOperation.getAseRPCName ());
        tempCmd.append (" (");
        if (_bindingStyle.value == 0)
        {
            boolean needComma = false;
            // Output  parameters
            Vector outParms = aOperation.getOutputParameters ();
            Iterator iParms = null;
            ase2parameter aParm = null;
            aParm = aOperation.getReturnParmeter ();
            if (aParm != null)
            {
                tempCmd.append (addColumn (aParm, false, false));
                needComma = true;
            }
            if (outParms != null)
            {
                iParms = outParms.iterator ();
                while (iParms.hasNext ())
                {
                    aParm = (ase2parameter) iParms.next ();
                    tempCmd.append (addColumn (aParm, needComma, false));
                    needComma = true;
                }
            }
            // Input parameters
            Vector inParms = aOperation.getInputParameters ();
            if (inParms != null)
            {
                iParms = inParms.iterator ();
                aParm = null;
                while (iParms.hasNext ())
                {
                    aParm = (ase2parameter) iParms.next ();
                    tempCmd.append (addColumn (aParm, needComma, true));
                    needComma = true;
                }
            }
        }
```

```java
            else if (_bindingStyle.value == 1)
            {
                tempCmd.append ("outxml varbinary(15000), _inxml varchar("
                + Globals.pageSize + ") null");
                // tempCmd.append("outxml varchar(" + Globals.pageSize + "), _inxml varchar(" + Globals.pageSize + ")
null");
                // tempCmd.append("outxml text, _inxml varchar(" + Globals.pageSize + ") null");
            }
            // String cisRemote = System.getProperty (CIS_REMOTE);
            String cisRemote = service + ".XMLCONNECT";
            tempCmd.append (" ) external procedure at '" + cisRemote + ".");
            tempCmd.append (_aseOwnerName);
            tempCmd.append (".");
            tempCmd.append (aOperation.getAseRPCName ());
            tempCmd.append ("'");
            tempCmd.append ("\n");
        }
        cisCommand.append (tempCmd.toString ());
    }
    return (cisCommand.toString ());
}
public String addColumn (ase2parameter aParm, boolean needComma, boolean input)
{
    StringBuffer tempBuffer = new StringBuffer ("");
    String colName = aParm.getColumnName ();
    if (needComma == true)
    {
        tempBuffer.append (", ");
    }
    if (input == false)
    {
        if (colName.charAt (0) == '_')
        {
            tempBuffer.append (aseutil.mapIdentifier (colName.substring (1)));
        }
        else
        {
            tempBuffer.append (aseutil.mapIdentifier (colName));
        }
    }
    else
    {
        if (colName.charAt (0) == '_')
        {
            tempBuffer.append (colName);
        }
        else
        {
            tempBuffer.append ("_" + aseutil.mapIdentifier (colName));
        }
    }
```

```java
        }
        tempBuffer.append (" ");
        tempBuffer.append (aParm.getASEType ());
        if (input == true)
        {
            tempBuffer.append (" null ");
        }
        return(tempBuffer.toString ());
    }
}
// ase2wsdl.java
// Copyright (c) 2003. Sybase, Inc. All Rights Reserved.
//
//Confidential property of Sybase, Inc.
//Copyright 1987, 2003
//Sybase, Inc.  All rights reserved.
//Unpublished rights reserved under U.S. copyright laws.
//
//
package com.sybase.ase.ws.server;
/**
 * FileName:   ase2wsdl.java
 * PackageName: com.sybase.ase.ws.server
 *
 * Description:
 * This class represents a WSDL file and provides all the information
 * need to invoke operations in it from ASE.
 *
 */
import java.util.Vector;
import java.util.Iterator;
import java.util.Hashtable;
import org.apache.axis.wsdl.symbolTable.SymbolTable;
import org.apache.axis.wsdl.symbolTable.ServiceEntry;
import javax.wsdl.Service;
import java.net.URL;
import java.io.File;
import java.net.URLClassLoader;
import java.net.MalformedURLException;
import com.sybase.ase.ws.util.Globals;
public class ase2wsdl
{
    String  _jarFileName;
    String  _fullJarFileName;
    ServiceEntry _sEntry;
    Service  _service;
    SymbolTable _symTable;
    Vector  _services;
    URLClassLoader _jarLoader;
    public ase2wsdl (ServiceEntry sEntry, SymbolTable symTable)
```

```java
{
    _jarFileName = sEntry.getName () + ".jar";
    _sEntry = sEntry;
    _symTable = symTable;
    _services = new Vector ();
    ase2service aseService = new ase2service (this, _sEntry, _symTable);
    _services.add (aseService);
    String wsdlURI = symTable.getWSDLURI ();
    Globals.webMethods.put (wsdlURI, this);
}
public void setFullJarName (String fullJarFileName)
{
    _fullJarFileName = fullJarFileName;
}
public URLClassLoader getJarLoader ()
{
    if (_jarLoader == null)
    {
        URL[] dummy = new URL[1];
        try
        {
            dummy[0] = new File (_fullJarFileName).toURL ();
            _jarLoader = new URLClassLoader (dummy);
            Globals.xmlLogger.info ("making jar loader success with: "
            + dummy[0]);
        }
        catch (MalformedURLException err)
        {
            Globals.xmlLogger.fatal ("Cuaght exception: " + err);
        }
    }
    return (_jarLoader);
}
public String genCISSql (String service)
{
    String cissql = "";
    Iterator allServices = _services.iterator ();
    ase2service aService = null;
    while (allServices.hasNext ())
    {
        aService = (ase2service) allServices.next ();
        cissql += aService.genCIS (service);
    }
    return (cissql);
}
public Hashtable getMappings ()
{
    Hashtable toReturn = new Hashtable ();
    Iterator allServices = _services.iterator ();
    ase2service aService = null;
```

```java
        while (allServices.hasNext ())
        {
            aService = (ase2service) allServices.next ();
            Iterator allOperations = aService.getOperations ().iterator ();
            ase2operation aOperation = null;
            while (allOperations.hasNext ())
            {
                aOperation = (ase2operation) allOperations.next ();
                toReturn.put (aOperation.getOperationName (),
                aOperation.getAseRPCName ());
            }
        }
        return (toReturn);
    }
    public String getKey ()
    {
        return (_sEntry.getName ());
    }
    public String getJarFileName ()
    {
        return (_jarFileName);
    }
    public Vector getServices ()
    {
        return (_services);
    }
}
// XMLRpcHandler.java
// Copyright (c) 2003. Sybase, Inc. All Rights Reserved.
//
//Confidential property of Sybase, Inc.
//Copyright 1987, 2003
//Sybase, Inc.  All rights reserved.
//Unpublished rights reserved under U.S. copyright laws.
//
//
package com.sybase.ase.ws.sds;
/**
 * FileName:   XMLRpcHandler.java
 * PackageName: com.sybase.ase.ws.sds
 *
 * Description:
 * Handler for all RPC events from ASE/CIS.
 *
 *
 */
import java.sql.ResultSet;
import com.sybase.jdbc2.tds.SrvSession;
import com.sybase.jdbc2.tds.SrvDbrpcToken;
import com.sybase.jdbc2.tds.SrvDataFormat;
```

```java
import com.sybase.jdbc2.tds.TdsConst;
import com.sybase.jdbc2.tds.SrvLoginToken;
//import com.sybase.jdbc2.tds.SrvRowDataFormat2;
import java.util.StringTokenizer;
import com.sybase.ase.ws.server.ase2wsdl;
import com.sybase.ase.ws.server.ase2operation;
import com.sybase.ase.ws.server.aseutil;
import com.sybase.ase.ws.server.ase2service;
import com.sybase.ase.ws.server.opReturn;
import com.sybase.ase.ws.util.Globals;
import com.sybase.ase.ws.util.LogConstants;
import com.sybase.ase.ws.util.Utility;
import com.sybase.ase.ws.axis.Wsdl2ase;
import java.util.Enumeration;
import java.util.Vector;
import java.util.Iterator;
// import java.util.Hashtable;
import java.sql.Connection;
import java.sql.SQLException;
import java.io.IOException;
import java.io.File;
public class XMLRpcHandler implements LogConstants, SDSConstants
{
    /**
     * @param receiver Receiver that allows us to send a TDS_DONE
     * @param s TDS session information
     * @param rpc RPC information
     * @param o RPC arguments.
     * @throws IOException
     *
     * Description:
     * Entry point from XMLReciever class for RPC events.
     *
     */
    public void handleRPC (XMLReceiver receiver,
    SrvSession s,
    SrvDbrpcToken rpc,
    Object[] o) throws IOException
    {
        Globals.xmlLogger.trace ("XMLRpcHandler.handleRPC");
        try
        {
            // Grab the name of the rpc and figure out what to
            // do with it.
            String rpcName = rpc.getName ();
            int returnStatus = 0;
            boolean done = true;
            Globals.xmlLogger.info ("RPC Name is: " + rpcName);
            // If the rpc name is known to us as a web method,
            // we process it.
```

```java
        ase2operation operation = isWebMethod (rpcName);
        if (operation != null)
        {
            handleWebMethod (receiver, s, rpc, o, operation);
        } // No-op this RPC as passthru mode is meaningless to us for now.
        else if (rpcName.equalsIgnoreCase (SP_THREAD_PROPS))
        {
            returnStatus = 0;
            done = true;
            receiver.sendRPCParams (s, returnStatus, null, null, done);
        } // We have to support SP_CAPABILITES for CIS.
        // But we don't do much of anything.
        else if (rpcName.equalsIgnoreCase (SP_CAPABILITIES))
        {
            // sendRPCParms is called from within handleSPcapabilities.
            handleSPcapabilities (receiver, s, rpc, o);
        } // Generate a proxy table given a WSDL file here.
        else if (rpcName.equalsIgnoreCase (GEN_SPROC_FROM_WSDL)
        || rpcName.equalsIgnoreCase (SP_GEN_SPROC_FROM_WSDL))
        {
            // sendRPCParms is called from within handleSproc.
            handleSproc (receiver, s, rpc, o);
        }
        else if (rpcName.equalsIgnoreCase (GET_MAPPINGS))
        {
            // sendRPCParms is called from within handleSproc.
            handleMappings (receiver, s, rpc, o);
        } // Otherwise send back a note that says we failed.
        else
        {
            Globals.xmlLogger.info ("unknown rpc " + rpcName);
            returnStatus = 0;
            done = true;
            receiver.sendMessage (s, 32000, "Unknown procedure", rpcName, 1);
            receiver.SendDone (s, -1, true, true, true);
            // receiver.sendRPCParams(s, returnStatus, null, null, done);
        }
    }
    catch (IOException err)
    {
        Globals.xmlLogger.fatal (err.getMessage ());
        Globals.xmlLogger.fatal ("IOException from SendDone in handleRPC");
        receiver.SendDone (s, 1, false, true, true);
    } // If any of the handling erred in communincating with ASE
    // (primarily is this in GEN_SPROC_FROM_WSDL, bail out.
    catch (SQLException err)
    {
        Globals.xmlLogger.fatal (err.getMessage ());
        Globals.xmlLogger.fatal ("IOException from SendDone in handleRPC");
        receiver.SendDone (s, 1, false, true, true);
```

```java
        }
        Globals.xmlLogger.trace ("XMLRpcHandler.handleRPC");
    }
    /**
     * @param rpcName Name of the RPC
     * @return ase2operation
     *
     * Description:
     * If the RPC is of the type name.name.name we look to see
     * if it a web method mapped to a proxy table.
     *
     */
    protected ase2operation isWebMethod (String rpcName)
    {
        ase2operation toReturn = null;
        String[] items = Utility.split (rpcName, "\\.");
        if (items.length == 3)
        {
            toReturn = findOperation (items[1], items[2]);
        }
        return (toReturn);
    }
    /**
     * @param receiver
     * @param s
     * @param rpc
     * @param o
     * @param operation
     * @throws IOException
     *
     * Description:
     * Things get interesting here. We have a request to invoke
     * a web methods. So off we go....
     *
     */
    protected void handleWebMethod (XMLReceiver receiver,
    SrvSession s,
    SrvDbrpcToken rpc,
    Object[] o,
    ase2operation operation) throws IOException
    {
        Globals.xmlLogger.trace ("XMLRpcHandler.handleWebMethod");
        Globals.xmlLogger.trace ("handling rcp name: " + rpc.getName ());
        Globals.xmlLogger.trace ("operation name is: "
        + operation.getAseRPCName ());
        boolean error = false;
        int count = 0;
        opReturn toReturn = null;
        // Invoke the web method.
        try
```

```java
        {
            // It's possible we don't have args, but have a
            // return value.
            if (rpc.hasParams ())
            {
                toReturn = operation.invoke (o);
            }
            else
            {
                toReturn = operation.invoke (null);
            }
            // Now try and send the data back to ASE. We send back a set
            // of Rows because that is what CIS expects and not TDS_PARMFMT.
            if (toReturn != null)
            {
                receiver.sendResults (s, toReturn.desc, toReturn.data);
                count = toReturn.data.length;
            }
            else
            {
                count = 0;
            }
        }
        catch (SQLException err)
        {
            receiver.createMessage (s, 12, err.getMessage (),
            "ASE Web Services sds", rpc.getName (), 1);
            dumpException (err);
        }
        catch (Exception err)
        {
            receiver.createMessage (s, 12, err.getMessage (),
            "ASE Web Services sds", rpc.getName (), 1);
            dumpException (err);
        }
        receiver.SendDone (s, count, error, true, true);
        Globals.xmlLogger.trace ("XMLRpcHandler.handleWebMethod");
}
/**
 * @param service
 * @param operation
 * @return ase2operation
 *
 * Description:
 * Loop through our list of know web method operations,
 * to see if we have this one.
 *
 */
protected ase2operation findOperation (String service, String operation)
{
```

```java
        ase2operation toReturn = null;
        Enumeration eMethods = Globals.webMethods.elements ();
        ase2wsdl aWSDL = null;
        Vector vServices = null;
        Iterator iServices = null;
        ase2service aService = null;
        Vector vOperations = null;
        Iterator iOperations = null;
        ase2operation aOperation = null;
        while (eMethods.hasMoreElements ())
        {
            aWSDL = (ase2wsdl) eMethods.nextElement ();
            vServices = aWSDL.getServices ();
            iServices = vServices.iterator ();
            while (iServices.hasNext ())
            {
                aService = (ase2service) iServices.next ();
                if (service.compareToIgnoreCase (aService.getAseOwnerName ())
                == 0)
                {
                    vOperations = aService.getOperations ();
                    iOperations = vOperations.iterator ();
                    while (iOperations.hasNext ())
                    {
                        aOperation = (ase2operation) iOperations.next ();
                        if (operation.compareTo (aOperation.getAseRPCName ())
                        == 0)
                        {
                            toReturn = aOperation;
                            break;
                        }
                    }
                }
            }
        }
        return (toReturn);
    }
    /**
     * @param receiver
     * @param s
     * @param rpc
     * @param o
     * @throws IOException
     * @throws SQLException
     *
     * Description:
     * The meat of the work for turning a WSDL file into
     * a proxy table is done here.
     *
     */
```

```java
protected void handleMappings (XMLReceiver receiver,
SrvSession s,
SrvDbrpcToken rpc,
Object[] o) throws IOException, SQLException
{
    Globals.xmlLogger.trace ("XMLRpcHandler.handleMappings");
    int returnStatus = 1;
    boolean done = true;
    Object[] desc = null;
    String sqlCommands = null;
    if (rpc.hasParams () == false)
    {
        returnStatus = 1;
    }
    else
    {
        try
        {
            Connection conn = null;
            SrvLoginToken sLoginToken = s.getLogin();
            String userName = sLoginToken.getUser();
            String password = sLoginToken.getPassword();
            String service = sLoginToken.getHost();
            generateWSDLMapping (s, rpc, o);
            String wsdlFile = (String) o[0];
            String aseServerName = Globals.si.getHostName (service);
            Globals.xmlLogger.debug ("aseServerName is: " + aseServerName);
            String asePortStr = Globals.si.getPortNumber (service);
            Globals.xmlLogger.debug ("asePortStr is: " + asePortStr);
            if (aseServerName != null && asePortStr != null)
            {
                int asePortNumber = Integer.parseInt(asePortStr);
                conn = aseutil.login (aseServerName, asePortNumber, null,
                userName, password);
                if (conn != null)
                {
                    if (Globals.pageSize == PAGESIZE_UNINITIALIZED)
                    {
                        ResultSet rs = aseutil.executeStatement (conn,
                        "select @@pagesize");
                        // Move to the first row.
                        rs.next ();
                        Globals.pageSize = rs.getInt (1);
                    }
                    // Setup to parse the wsdl file.
                    Wsdl2ase myWsdl = new Wsdl2ase ();
                    myWsdl.addFactory ();
                    String args[] = new String[3];
                    // What directory should we use for generating the
                    // stub java files and compiling them?
```

```java
String rootDir;
rootDir = System.getProperty (TEMPDIR_PROPERTY);
if (rootDir == null)
{
    rootDir = System.getProperty ("java.io.tmpdir")
    + "xmlconnect";
}
boolean dirCreated;
File rootDirFile = new File (rootDir);
if (rootDirFile.isDirectory () == false)
{
    dirCreated = rootDirFile.mkdirs ();
}
String jarDirectory = rootDir + File.separator + "jars";
File jarDirFile = new File (jarDirectory);
if (jarDirFile.isDirectory () == false)
{
    dirCreated = jarDirFile.mkdirs ();
}
String wsdlRootDir = rootDir + File.separator
+ System.currentTimeMillis ();
args[0] = "-o" + wsdlRootDir;
args[1] = "-O-1";
args[2] = wsdlFile;
// Launch the WSDL parsing. Note the WSDL can be located
// at a URL, so this parsing is done under a timeout condition.
// that throws a general Exception. The timeout condition is implemented
// by spawning a thread and killing it if a the timer fires. Note
// that Java timers are notoriously slow and nowhere near accurate.
// The best that can be said is that the timer will not fire before
// the specified time....
myWsdl.launch (args);
// Now we have a mapping between the WSDL file and
// an ase2wsdl object.
ase2wsdl mapping = (ase2wsdl) Globals.webMethods.get (wsdlFile);
// Generate the sql commands needed to create the proxy
// table.
sqlCommands = mapping.genClSSql (s.getLogin ().getServiceName ());
Globals.xmlLogger.info ("sqlcommands: <" + sqlCommands + ">");
// Compile the stub files so that we can use them when
// we invoke the web method.
Utility.compileDir (wsdlRootDir);
// Load the compiled class files and source files into
// a jar and modify the classpath as needed.
String jarFileName = jarDirectory + File.separator
+ mapping.getJarFileName ();
Utility.createJar (jarFileName, wsdlRootDir, true);
Utility.addJarToClasspath (jarFileName);
mapping.setFullJarName (jarFileName);
returnStatus = 0;
```

```java
            }
            else
            {
                Globals.xmlLogger.error ("Connection to ASE not created successfully");
            }
        }
        else
        {
            Globals.xmlLogger.error ("Invalid asePort or aseServer");
        }
    }
    catch (ClassNotFoundException err)
    {
        Globals.xmlLogger.fatal (err.getMessage ());
        returnStatus = 1;
    }
    catch (SQLException err)
    {
        Globals.xmlLogger.fatal (err.getMessage ());
        returnStatus = 1;
    }
    catch (Exception err)
    {
        Globals.xmlLogger.fatal (err.getMessage ());
        returnStatus = 1;
    }
}
if (returnStatus == 0)
{
    SrvDataFormat[] datadesc = new SrvDataFormat[1];
    datadesc[0] = new SrvDataFormat ("outparam", TdsConst.VARCHAR,
    TdsConst.ROW_UPDATABLE | TdsConst.ROW_NULLALLOWED, 255, null);
    int numRows = 0;
    StringTokenizer st = new StringTokenizer (sqlCommands, "\n", false);
    String token = null;
    while (st.hasMoreTokens ())
    {
        token = st.nextToken ();
        numRows++;
    }
    Object[][] localData = new Object[numRows][1];
    st = null;
    st = new StringTokenizer (sqlCommands, "\n", false);
    int index = 0;
    while (st.hasMoreTokens ())
    {
        token = st.nextToken();
        localData[index][0] = new String (token);
        index++;
    }
```

```java
        int count = receiver.sendResults (s, datadesc, localData);
        receiver.SendDone (s, count, false, true, true);
    }
    else
    {
        // Something failed, we send nothing back, but we have to
        // tell CIS that we're done.
        // perhaps, we should send an EED here.
        receiver.sendRPCParams (s, 0, null, null, true);
    }
    Globals.xmlLogger.trace ("XMLRpcHandler.handleMappings");
}
protected void generateWSDLMapping (SrvSession s,
SrvDbrpcToken rpc,
Object[] o)
{}
/**
 * @param receiver
 * @param s
 * @param rpc
 * @param o
 * @throws IOException
 * @throws SQLException
 *
 * Description:
 * The meat of the work for turning a WSDL file into
 * a proxy table is done here.
 *
 */
protected void handleSproc (XMLReceiver receiver,
SrvSession s,
SrvDbrpcToken rpc,
Object[] o) throws IOException, SQLException
{
    Globals.xmlLogger.trace ("XMLRpcHandler.handleSproc");
    int returnStatus = 0;
    boolean done = true;
    Object[] desc = null;
    String sqlCommands = null;
    if (rpc.hasParams () == false)
    {
        returnStatus = 1;
    }
    else
    {
        try
        {
            Connection conn = null;
            SrvLoginToken sLoginToken = s.getLogin();
            String userName = sLoginToken.getUser();
```

```java
String password = sLoginToken.getPassword();
String service = sLoginToken.getHost();
generateWSDLMapping (s, rpc, o);
String wsdlFile = (String) o[0];
String aseServerName = Globals.si.getHostName (service);
Globals.xmlLogger.debug ("aseServerName is: " + aseServerName);
String asePortStr = Globals.si.getPortNumber (service);
Globals.xmlLogger.debug ("asePortStr is: " + asePortStr);
if (aseServerName != null && asePortStr != null)
{
    int asePortNumber = Integer.parseInt(asePortStr);
    conn = aseutil.login (aseServerName, asePortNumber, null,
    userName, password);
    if (conn != null)
    {
        if (Globals.pageSize == PAGESIZE_UNINITIALIZED)
        {
            ResultSet rs = aseutil.executeStatement (conn,
            "select @@pagesize");
            // Move to the first row.
            rs.next ();
            Globals.pageSize = rs.getInt (1);
        }
        // Setup to parse the wsdl file.
        Wsdl2ase myWsdl = new Wsdl2ase ();
        myWsdl.addFactory ();
        String args[] = new String[3];
        // What directory should we use for generating the
        // stub java files and compiling them?
        String rootDir;
        rootDir = System.getProperty (TEMPDIR_PROPERTY);
        if (rootDir == null)
        {
            rootDir = System.getProperty ("java.io.tmpdir")
            + "xmlconnect";
        }
        boolean dirCreated;
        File rootDirFile = new File (rootDir);
        if (rootDirFile.isDirectory () == false)
        {
            dirCreated = rootDirFile.mkdirs ();
        }
        String jarDirectory = rootDir + File.separator + "jars";
        File jarDirFile = new File (jarDirectory);
        if (jarDirFile.isDirectory () == false)
        {
            dirCreated = jarDirFile.mkdirs ();
        }
        String wsdlRootDir = rootDir + File.separator
        + System.currentTimeMillis ();
```

```java
            args[0] = "-o" + wsdlRootDir;
            args[1] = "-O-1";
            args[2] = wsdlFile;
            // Launch the WSDL parsing. Note the WSDL can be located
            // at a URL, so this parsing is done under a timeout condition.
            // that throws a general Exception. The timeout condition is implemented
            // by spawning a thread and killing it if a the timer fires. Note
            // that Java timers are notoriously slow and nowhere near accurate.
            // The best that can be said is that the timer will not fire before
            // the specified time....
            myWsdl.launch (args);
            // Now we have a mapping between the WSDL file and
            // an ase2wsdl object.
            ase2wsdl mapping = (ase2wsdl) Globals.webMethods.get (wsdlFile);
            // Generate the sql commands needed to create the proxy
            // table.
            sqlCommands = mapping.genCISSql (s.getLogin ().getServiceName ());
            Globals.xmlLogger.info ("sqlcommands: <" + sqlCommands + ">");
            // Login to ASE and create the proxy tables only if we are asked to.
            aseutil.executeStatements (conn, sqlCommands);
            // Compile the stub files so that we can use them when
            // we invoke the web method.
            Utility.compileDir (wsdlRootDir);
            // Load the compiled class files and source files into
            // a jar and modify the classpath as needed.
            String jarFileName = jarDirectory + File.separator
                 + mapping.getJarFileName ();
            Utility.createJar (jarFileName, wsdlRootDir, true);
            Utility.addJarToClasspath (jarFileName);
            mapping.setFullJarName (jarFileName);
         }
      }
   }
   catch (ClassNotFoundException err)
   {
      Globals.xmlLogger.fatal (err.getMessage ());
      returnStatus = 1;
   }
   catch (SQLException err)
   {
      Globals.xmlLogger.fatal (err.getMessage ());
      returnStatus = 1;
   }
   catch (Exception err)
   {
      Globals.xmlLogger.fatal (err.getMessage ());
      returnStatus = 1;
   }
}
if (returnStatus == 0)
```

```java
    {
        SrvDataFormat[] datadesc = new SrvDataFormat[1];
        datadesc[0] = new SrvDataFormat ("outparam", TdsConst.VARCHAR,
        TdsConst.ROW_UPDATABLE | TdsConst.ROW_NULLALLOWED, 255, null);
        int numRows = 0;
        StringTokenizer st = new StringTokenizer (sqlCommands, "\n", false);
        String token = null;
        while (st.hasMoreTokens ())
        {
            token = st.nextToken ();
            numRows++;
        }
        Object[][] localData = new Object[numRows][1];
        st = null;
        st = new StringTokenizer (sqlCommands, "\n", false);
        int index = 0;
        while (st.hasMoreTokens ())
        {
            token = st.nextToken();
            localData[index][0] = new String (token.substring(0, (token.length() - 1)));
            index++;
        }
        int count = receiver.sendResults (s, datadesc, localData);
        receiver.SendDone (s, count, false, true, true);
    }
    else
    {
        // Something failed, we send nothing back, but we have to
        // tell CIS that we're done.
        // perhaps, we should send an EED here.
        receiver.sendRPCParams (s, 0, null, null, true);
    }
    Globals.xmlLogger.trace ("XMLRpcHandler.handleSproc");
}
/**
 * @param receiver
 * @param s
 * @param rpc
 * @param o
 * @throws IOException
 * @throws SQLException
 *
 * Description:
 * CIS invokes this stored procedure to find our capabilites.
 * We don't do much of anything.
 *
 */
protected void handleSPcapabilities (XMLReceiver receiver,
SrvSession s,
SrvDbrpcToken rpc,
```

```java
Object[] o) throws IOException, SQLException
{
    Globals.xmlLogger.trace ("XMLRpcHandler.handleSPcapabilities");
    SrvDataFormat[] desc = new SrvDataFormat[3];
    desc[0] = new SrvDataFormat ("id", TdsConst.INT4, TdsConst.ROW_UPDATABLE,
    4, null);
    desc[1] = new SrvDataFormat ("capability", TdsConst.CHAR,
    TdsConst.ROW_UPDATABLE, 30, null);
    desc[2] = new SrvDataFormat ("value", TdsConst.INT4,
    TdsConst.ROW_UPDATABLE, 4, null);
    Object[][] localData = {
        {
            new Integer (101), "sql syntax", new Integer (0)
        }, {
            new Integer (102), "join handling", new Integer (0)
        }, {
            new Integer (103), "aggregate handling", new Integer (0)
        }, {
            new Integer (104), "AND predicates", new Integer (0)
        }, {
            new Integer (105), "OR predicates", new Integer (0)
        }, {
            new Integer (106), "LIKE predicates", new Integer (0)
        }, {
            new Integer (107), "bulk insert handling", new Integer (0)
        }, {
            new Integer (108), "text/image handling", new Integer (1)
        }, {
            new Integer (109), "transaction handling", new Integer (0)
        }, {
            new Integer (110), "textpattern handling", new Integer (0)
        }, {
            new Integer (111), "order by", new Integer (0)
        }, {
            new Integer (112), "group by", new Integer (0)
        }, {
            new Integer (113), "new password encryption", new Integer (0),
        },
        {
            new Integer (114), "object name case sensitivity",
            new Integer (0),
        }, {
            new Integer (115), "distinct handling", new Integer (0),
        }, {
            new Integer (117), "union support", new Integer (0),
        }, {
            new Integer (118), "string functions", new Integer (0),
        }, {
            new Integer (119), "expression handling", new Integer (0),
        }, {
```

```java
            new Integer (120), "truncate blanks", new Integer (0),
        }, {
            new Integer (121), "language handling", new Integer (0),
        }, {
            new Integer (122), "date functions", new Integer (0),
        }, {
            new Integer (123), "math functions", new Integer (0),
        }, {
            new Integer (124), "convert function", new Integer (0),
        }, {
            new Integer (125), "tsql delete/update", new Integer (0),
        }, {
            new Integer (126), "insert select", new Integer (0),
        }, {
            new Integer (127), "subquery support", new Integer (0),
        }, {
            new Integer (0), "", new Integer (0)
        }
    };
    receiver.sendResults (s, desc, localData);
    receiver.SendDone (s, 0, false, true, true);
    Globals.xmlLogger.trace ("XMLRpcHandler.handleSPcapabilities");
}
/**
 * @param err
 *
 * Description:
 *
 * Notes:
 * Convert the exception into something that can be put into
 * the log file.
 *
 */
private void dumpException (Exception err)
{
    java.io.CharArrayWriter charArray = new java.io.CharArrayWriter ();
    java.io.PrintWriter getTrace = new java.io.PrintWriter (charArray);
    err.printStackTrace (getTrace);
    getTrace.flush ();
    Globals.xmlLogger.fatal (charArray.toString ());
}
}
```